

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-224951
(43)Date of publication of application : 03.09.1993

(51)Int.CI. G06F 9/46
G06F 13/24

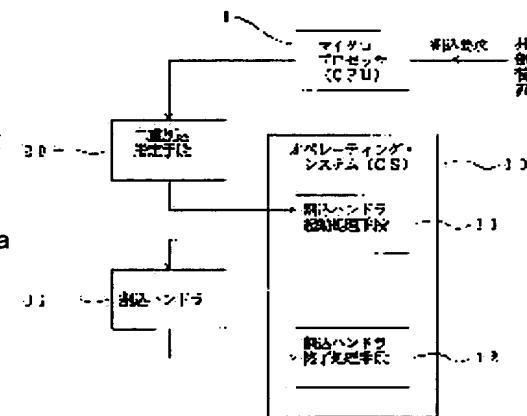
(21)Application number : 04-024226 (71)Applicant : FUJITSU COMMUN SYST LTD
(22)Date of filing : 12.02.1992 (72)Inventor : NAKAJIMA YASUO
YAMAWAKI TOMOHIRO
ATAMI NOBUO

(54) INTERRUPTION MANAGING METHOD FOR MICROPROCESSOR SYSTEM

(57)Abstract:

PURPOSE: To provide the interruption managing method for generating easily an interruption handler, and decreasing an overhead of an OS.

CONSTITUTION: In the microprocessor system for suspending a processing at the time when a CPU 1 which is executing a processing through an OS 10, receives an interruption request from an external device, and allowing an interruption handler 31 to execute an interruption processing, this system is constituted by providing a double interruption designating means 20 for designating a soft interruption at the time when it is actuated from the CPU which receives the interruption request and migrating the processing to the OS before migrating the processing to the interruption handler, an interruption handler actuation processing means 11 for analyzing an interruption factor and retaining a register, etc., in the OS and actuating the interruption handler, and an interruption handler finish-processing means 12 for executing reset of the register, etc., and an interruption finish processing to the external device in the OS at the time when an end of the interruption processing is informed from the interruption handler, and thereafter, restarting the suspended processing.



LEGAL STATUS

[Date of request for examination] 06.01.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C) 1998,2000 Japanese Patent Office

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平5-224951

(43)公開日 平成5年(1993)9月3日

(51)Int.Cl.⁵ 識別記号 庁内整理番号 F I 技術表示箇所
G 06 F 9/46 310 Z 8120-5B
13/24 310 Z 9072-5B

審査請求 未請求 請求項の数1(全14頁)

(21)出願番号 特願平4-24226

(22)出願日 平成4年(1992)2月12日

(71)出願人 000237651

富士通コミュニケーション・システムズ株式会社
神奈川県横浜市港北区新横浜3丁目9番18号

(72)発明者 中島 康夫

神奈川県横浜市港北区新横浜3丁目9番18号 富士通コミュニケーション・システムズ株式会社内

(74)代理人 弁理士 井桁 貞一

最終頁に続く

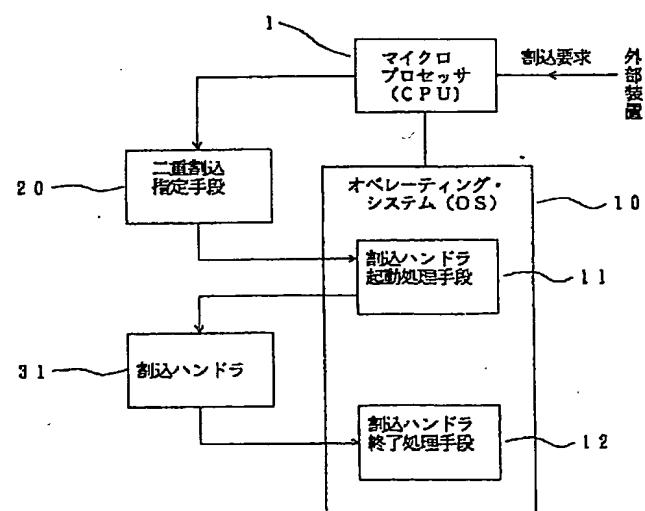
(54)【発明の名称】 マイクロプロセッサ・システムの割込管理方法

(57)【要約】

【目的】 マイクロプロセッサ・システムの割込管理方法に関し、割込ハンドラの作成が容易で、OSのオーバーヘッドが少ない割込管理方法を提供することを目的とする。

【構成】 OS10を介して処理実行中のCPU1が外部装置より割込要求を受けたときに処理を中断して割込ハンドラ31に割込処理を行わせるマイクロプロセッサ・システムにおいて、割込要求を受けたCPUより起動されたときにソフト割込を指定して割込ハンドラへの処理移行前にOSに処理を移行させる二重割込指定手段20と、二重割込指定手段を介して処理を移行されたとき、OS内で割込要因の解析とレジスタ類の保存を行って割込ハンドラを起動する割込ハンドラ起動処理手段11と、割込ハンドラより割込処理の終了を通知されたときにOS内でレジスタ類の復帰と外部装置に対する割込終了処理を行ったのちに中断した処理を再開させる割込ハンドラ終了処理手段12を備えるように構成する。

本発明の基本構成図



【特許請求の範囲】

【請求項1】 オペレーティングシステム(10)を介してプログラム処理を実行中のマイクロプロセッサ(1)が外部装置より割込要求を受けたときに実行中のプログラム処理を中断して割込要因ごとの処理を規定する割込ハンドラ(31)に割込処理を行わせるマイクロプロセッサ・システムにおいて、

割込要求を受けた前記マイクロプロセッサ(1)より起動されたときにソフト割込を指定して前記割込ハンドラ(31)への処理移行前に前記オペレーティングシステム(10)に処理を移行させる二重割込指定手段(20)と、

前記オペレーティングシステム(10)内において、前記二重割込指定手段(20)を介して処理を移行されたときに、割込要因の解析と前記マイクロプロセッサ(1)の内部レジスタ類の保存を行ったのち、前記割込ハンドラ(31)を起動する割込ハンドラ起動処理手段(11)と、

前記オペレーティングシステム(10)内において、前記割込ハンドラ(31)より割込処理の終了を通知されたときに、前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる割込ハンドラ終了処理手段(12)を備えたことを特徴とするマイクロプロセッサ・システムの割込管理方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明はマイクロプロセッサ・システムの割込管理方法に関する。近年、多くのマイクロプロセッサ・システムが使用されているが、マイクロプロセッサ・システムにおいてはマイクロプロセッサ(以下、CPUと記す)がプログラムの処理を実行中に外部デバイスまたは外部装置(以下、外部装置と総称する)から割込要求が行われたときに実行中のプログラム処理を中断して外部装置の割込要求を処理することが多い。

【0002】 外部要求で処理する内容は外部装置ごとに異なるので、ユーザが外部装置ごとに割込処理ルーチン(以下、割込ハンドラと記す)を作成しておき、割込要求があったときにCPUが割込要求を行った外部装置を識別し、該当する割込ハンドラを呼び出して処理させる方法が用いられている。

【0003】 外部割込の発生によりCPUが実行中のプログラム処理を中断する際には、再開するプログラムの番地や、それまで使用していたCPUの内部レジスタの内容を保存してから割込処理を実行し、割込処理が終了したときにこれらのレジスタを元の状態に復してプログラムを再開するが、前記割込ハンドラの中で内部レジスタの退避処理を記述することは割込ハンドラの作成を煩雑にするため、これらの処理はOS(オペレーティング・システム)の割込管理機能によって行うのが普通である。

【0004】 しかし、現在普及しているマイクロプロセ

ッサ・システムでは割込要求を受けたCPUがハード的に割込ハンドラを起動するようになっているため、もしそのまま割込ハンドラによって割込処理が開始されるとOSは割り込みの発生を知ることができず、レジスタ類の保存を行うこともできなくなる。このため、割込ハンドラ側からOSを呼び出して(システムコールと呼ぶ)OSにレジスタの保存などの処理を行ってもらい、これらの処理の終了後に再び割込ハンドラに処理を戻して割込処理を行う方法が用いられている。割込処理の終了時も同様に割込ハンドラからシステムコールを行い、OSがレジスタ等の復帰を行ったのち、中断していたプログラムを再開させる。なお、このシステムコールによってOSは割込処理の状態を管理することが可能となっている。

【0005】 一方、ユーザが作成する割込ハンドラは現在では高級言語によって書かれるのが普通であるが、高級言語によって前記のシステムコールを含めて記述すると、機械語に翻訳した時点で、モジュール(ハンドラ)の先頭で必ずスタックの操作と、いくつかのCPU内レジスタの書き替えが行われるようになっている。このため、CPUのレジスタは内容を退避する前に使用されることとなる。割込処理の終了時にも同様な問題が発生する。

【0006】 このような状態を避けるためにはシステムコールに前後する部分を高級言語でなくアセンブラー言語を用いて記述する必要がある。しかし、現在ではアセンブラー言語による記述は一般的でないため、正確な記述を行うのが難しく、加えて割込ハンドラの内部で分岐などがあると終了時の処理に誤りが生じ易い。

【0007】 このため、割込ハンドラを高級言語のみで記述することが可能で、処理誤りが発生し難いマイクロプロセッサ・システムの割込管理方法が必要となっている。

【0008】

【従来の技術】 図6はマイクロプロセッサ・システム構成図、図7は従来技術のシステム構成図、図8は従来技術のメモリ構成図、図9は従来技術の処理移行状態の説明図、図10は従来技術のフロー図である。

【0009】 図6は本発明が適用されるマイクロプロセッサ・システムの構成の一例を示している。図6のCPU1は8086シリーズのマイクロプロセッサ(CPU)を用いて構成されたマイクロプロセッサ・システムの例を示しているが、CPU1はバス6を介してRAM(ランダムアクセス・メモリ)4、ROM(読み専用メモリ)5、割込コントローラ(以下、PICと記す)2及び8台の外部装置(I/O)3-1~3-8と接続されている。

【0010】 PIC2は8台の外部装置3-1~3-8のうちの複数からIR₀~IR₇線を介して同時に割込要求があった場合に優先順位などを比較し、同時に一つの割

込要求のみがCPU1に行われるよう調停する役割をもち、割込要求はINTR線を通してCPU1に送出される。1つのPIC2には8台の外部装置が接続できるが、PIC2をカスケードに接続することにより最大256台までの外部装置を接続することができる。

【0011】図7は従来技術におけるマイクロプロセッサ・システムの割込処理関連部分のシステム構成を図示し、図8は図7の各部のメモリ（図7では図示省略）上の記憶状態の例を図示しているが、以下、図7及び図8により従来技術を説明する。なお、図7及び図8では2台の外部装置3-1～3-2のみを記載している。

【0012】図7の構成部分の一部は図8に示すようにRAMとROMにより構成されるメモリ上に記憶され、CPU1がこれらを読み出しながら各機能を動作させる。プログラム類はRAMまたはROMのいずれにも記憶可能なものが多いが、ここでは説明の便から、ROMにはOS60を構成する割込ハンドラ起動処理部61、割込ハンドラ終了処理部62及び実行管理部63のみが記憶され、割込ハンドラ81及びアプリケーション・プログラム（以下、APと記す）82はRAMに記憶されるものとする。なお、以下において単にOSと記した場合にはOS60全体またはその一部（主として実行管理部63）を指すものとする。

【0013】いま、CPU1がOSの実行管理部63との共同動作によりAP82の1200番地の命令を実行中であるとすると、CPU1内のプログラムカウンタ（以下、PCと記す）1aは図8に示すように実行中の番地の次の1201番地を指している。

【0014】この状態で外部装置3-1より割込要求があり、PIC2が他の割込要求と優先度などを比較してこの割込要求を優先することを確認すると、これをCPU1に伝える。割込要求の内容は外部装置ごとに異なるため、どの外部装置からの割込要求であるかを識別する必要があるが、以下、割込要求を行った外部装置を割込番号で識別することとし、外部装置3-1及び3-2の割込要求をそれぞれ割込番号1及び割込番号2とする。

【0015】図8の割込ベクタ領域71は各割込番号の割込処理内容を規定する割込ハンドラ81が記憶されているメモリ領域の先頭番地（ベクタと呼ばれる）を記憶しているメモリ領域であり、割込番号ごとに4バイト（セグメントとオフセット各2バイト）のアドレス情報を記憶している。

【0016】外部装置は最大256まで接続できるため、割込ベクタ領域71には0から255までの番号をもつ256の領域が確保されている。この番号をタイプナンバと呼び、各タイプナンバへの割り込みをタイプ0割込～タイプ255割込と呼んでいる。実際にはユーザが使用できるタイプナンバの範囲は限定されているが、ここでは説明の便から、割込番号1をタイプ0割込、割込番号2をタイプ1割込に対応させて説明する。

【0017】ベクタ領域71の物理アドレス（バイトアドレス）には例えば00000～003FF番地（16進数）が使用され、各タイプナンバごとに割り当てられた4バイトの先頭番地がアドレスとして使用される。従って、割込番号1は00000番地、割込番号2は00004番地を使用して割込ベクタ領域71にアクセスする。

【0018】外部装置3-1の割込要求がPIC2を介してCPU1に伝えられると、CPU1は図6のINTA線に割込要求を受け付けたことを知らせ、PIC2より割込要求元を識別する情報として前記タイプ0割込のアドレス情報“00000”が送られる。これによりCPU1は00000番地にアクセスし、その内容を読み取る。

【0019】タイプ0割込に記憶されている割込ハンドラの先頭番地が3000番地であるとすると、CPU1はこの番地、即ち、割込ハンドラ81の処理に移るが、それに先立ち、中断するAP82の情報を退避させる。

【0020】図8では退避先のメモリ領域はRAM上のスタック領域83に確保されており、CPU1内のスタックポインタ（以下、SPと記す）1bが退避情報を記憶させる番地を指している。退避する情報は割込処理が終了したときに復帰すべき番地であり、この場合はPC1aが指しているアドレスの“1201”となる。

【0021】CPU1がこの“1201”をスタック領域83に記憶させるとSP1bは次の番地に移動し、更に高位の割り込みに備える。これを終了したのち、PC1aは割込番号1用の割込ハンドラの先頭番地である3000番地に飛び、処理が割込ハンドラ81に移る。ここまで処理はOS60が関与せずにCPU1によって行われる。

【0022】割込ハンドラ81は処理を開始すると、先ず前述したシステムコールを行うのでCPU1はOS60の一部を構成する割込ハンドラ起動処理部61によりレジスタ類（図示省略）の退避、即ち、AP82で使用していたレジスタの内容の退避を開始する。

【0023】このとき、OS60は割込ハンドラ81から起動されたが、起動した割込ハンドラがどの外部装置のものであるか知らされていないため、PIC2にアクセスし、その内部レジスタ（図示省略）を読むことによって確認する。これによってOSは、現在どの割込処理が行われているかを管理することが可能となり、多重割込が行われた場合などに対処することができる。しかし、この割込要因の解析は、外部装置数が多く、PIC2がカスケード接続されているような場合には時間を要し、OSのオーバーヘッドが大きくなる原因となる。

【0024】内部レジスタ類の保存と割込要因の解析を終了すると、CPU1は再び処理の実行を割込ハンドラ81に移し、割込ハンドラ81によって実質的な割込処理が行われる。なお、割込ハンドラ81による割込処理の実行中に高位の割込が発生すると、CPU1及びOSは前記と同様な処理により高位の割込ハンドラを起動し、その割込処理を実行させるたのち、前の割込ハンドラに処理

を戻す。OSはこのような多重割込などに備え、常に割込状態を管理する。

【0025】割込処理が終了すると割込ハンドラ81は再びシステムコールをかけ、これによりOSの割込ハンドラ終了処理部62がレジスタ類の復帰やP I C 2の終了処理を行い、スタック領域83に記憶されているアドレス情報"1201"をP C1aに移すことにより中断していたAP82の処理が再開される。

【0026】以上においては説明を省略したが、現在のマイクロプロセッサ・システムが使用できるメモリ空間は極めて大きいため、アドレスの桁数が長大になることを避ける手段としてアドレスをセグメント（他と区別する場合はデータ・セグメントと呼ばれる）とオフセットに分け、セグメントを予めタスクごとに割り当てておく方法が使用されている。これにより、各プログラムはセグメントを意識することなく、オフセットのみを使用して作成することができる。しかし、アプリケーション・プログラムの処理から割込処理に移る場合にはセグメントが変わることとなるため、割り込みの際には割込ハンドラ81とOS60の間でセグメントの付け替えについての手順が取り決められる。

【0027】図9は割込処理の際の処理の移行状態を示しているが、図示のように、APの処理実行中に割込要求が発生すると、APの処理が中断されて割込ハンドラが起動される。割込ハンドラは起動されるとシステムコールによってOSを呼び出し、OSによるレジスタの保存などが終了するとOSより実行が移されて本来の割込処理を実行する。割込処理を終了すると再びシステムコールを行い、以後OSがレジスタの復帰などを行って中断していたAPの処理を再開する。

【0028】図10は以上におけるCPUとOSの処理のフロー図で、S11～S14及びS21～S23はフローの各ステップを示す記号である。図10の(1)は割込ハンドラ起動時のフローを示しているが、割込処理の開始時は図示のように、割込ベクタにより割込ハンドラを起動し(S11)、割込ハンドラよりのシステムコールによりOSの割込ハンドラ起動処理部がCPU内部レジスタの保存(S12)と割込要因の解析(S13)を行い、処理を割込ハンドラに移して(S14)処理を終了する。

【0029】図10の(2)は割込ハンドラ終了時のフローであり、割込ハンドラよりのシステムコールにより処理の終了を通知された割込ハンドラ終了処理部が終了処理を行うが、多重割込が行われていた場合はスタック領域に保存されていた中断中の割込ハンドラのアドレスを管理領域に復帰させ(S21)、P I C 2の終了処理(S22)と内部レジスタの復帰(S23)を行い、中断していた処理を再開する。

【0030】

【発明が解決しようとする課題】以上のように、従来技術ではOSがもつ割込要因解析機能、レジスタ保存機能

及び割込状態管理機能を用いてユーザが作成した割込ハンドラに割込処理を行わせているが、次のような問題を有している。

【0031】(1) ユーザが作成する割込ハンドラは起動後にシステムコールを行ってOSにレジスタの保存などをを行わせているが、割込ハンドラを高級言語で記述するとシステムコールの際に先ず処理に必要な数値を転送し、保存し終わっていないレジスタを使用する可能性があるため、高級言語のみで割込ハンドラを作成することができない。

【0032】(2) 割込ハンドラの中で分岐などが行われていると、終了処理が正確に行われず、終了時のシステムコールに誤りが発生し易いため、ユーザは割り込みの状態を意識して割込ハンドラを作成する必要がある。

【0033】(3) 割り込みの際にデータ・セグメントを変更する必要が生ずるため、OSとの間でデータ・セグメントの設定についての手順が必要となり、割込ハンドラの作成が複雑となる。

【0034】(4) 割込要因の解析の際にOSのオーバーヘッドが大きくなる。即ち、従来技術は割込ハンドラの作成が難しく、OSのオーバーヘッドが大きいという欠点を有している。

【0035】本発明は、割込ハンドラの作成が容易で、OSのオーバーヘッドが少ない割込管理方法を提供することを目的とする。

【0036】

【課題を解決するための手段】図1は本発明の基本構成図である。図中、1はマイクロプロセッサ(CPU)、10はオペレーティング・システム(OS)、31は外部割込における割込要因ごとの処理を規定する割込ハンドラ、20は外部装置(図示省略)より割込要求を受けた前記マイクロプロセッサ1より起動されたときにソフト割込を指定して前記割込ハンドラ31への処理移行前に前記オペレーティングシステム10に処理を移行させる二重割込指定手段である。

【0037】11は前記オペレーティングシステム10において、前記二重割込指定手段20を介して処理を移行されたときに、割込要因の解析と前記マイクロプロセッサ1の内部レジスタ類(図示省略)の保存を行ったのち、前記割込ハンドラ31を起動する割込ハンドラ起動処理手段、12は前記オペレーティングシステム10において、前記割込ハンドラ31より割込処理の終了を通知されたときに、前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる割込ハンドラ終了処理手段である。

【0038】

【作用】図1において、OS10を介してプログラム処理を実行中のCPU1が外部装置より割込要求を受けると、実行中のプログラム処理を中断して二重割込指定手段20にアクセスする。

【0039】二重割込指定手段20はCPU1に対して割込ハンドラ31が記憶されている番地を直接知らせずに、ソフト割込の実行を指定する。このため、外部装置よりの割込要求により直接割込ハンドラ31に処理が移されず、ソフト割込処理が開始される。

【0040】ソフト割込処理は外部割込に優先して行われるが、その際、ユーザが作成した割込ハンドラ31の代わりにOS10内部において割込処理の役割をもつ割込ハンドラ起動処理手段11が記憶されている番地を指定する。これにより、割込要求が行われたときに、CPU1のハード的な動作によって割込ハンドラ31に直接処理が移されずにOS10に処理が移される。

【0041】処理を移されたOS10内の割込ハンドラ起動処理手段11は割込要因の解析とCPU1の内部レジスタ類の保存を行ったのち、前記割込ハンドラ31を起動する。前記割込要因の解析によって割込要求を行った外部装置が識別され、以後、OSが割込状態を管理することが可能となるが、この割込要因の解析は外部装置側にアクセスすることなく簡単な方法で行われる（詳細後述）。

【0042】起動された割込ハンドラ31は割込処理を実行し、処理を終了するとOS10内の割込ハンドラ終了処理手段12に割込処理の終了を通知する。割込ハンドラ終了処理手段12は前記内部レジスタ類の復帰と前記外部装置に対する割込終了処理を行ったのち、中断した前記プログラム処理を再開させる。

【0043】以上のように、図1の構成においては外部割込が発生したときにCPU1が直接割込ハンドラ31に処理を移さずにOS10内の割込ハンドラ起動処理手段11に処理を移し、内部レジスタ類の保存と割込要因の解析を行ってから割込ハンドラ31を起動する。

【0044】このため、割込ハンドラ31はシステムコールによってレジスタの保存などを要求する必要がなくなり、割込処理の内容のみをすべて高級言語で作成することが可能となる。また、OSにおける割込要因の解析が簡単となるため、OSのオーバーヘッドが小さくなる。

【0045】

【実施例】図2は本発明の実施例のシステム構成図、図3は本発明の実施例のメモリ構成図、図4は本発明の実施例の処理移行状態を説明する図、図5は本発明の実施例のフロー図である。

【0046】全図を通じ、同一記号は同一対象物を示し、1はマイクロプロセッサ(CPU)、1aはプログラムカウンタ(PC)、1bはスタックポインタ(SP)、2は割込コントローラ(PIC)、3-1、3-2は外部装置、10はOS、11~13はOS10を構成する各部で、11は割込ハンドラ起動処理部、12は割込ハンドラ終了処理部、13は実行管理部である。

【0047】21、22は二重割込指定部20を構成する部分で、21は割込ベクタ領域、22は二重割込指定テーブルで

ある。また、31は割込ハンドラ、32はアプリケーション・プログラム(AP)、33はRAM上に設けらるスタック領域である。

【0048】図2はCPU1として8086シリーズのマイクロプロセッサを使用した本発明の実施例のマイクロプロセッサ・システムの構成図を示している。説明の便から図2及び図3には2台の外部装置3-1、3-2のみを記載している。以下、図2及び図3により本発明の実施例を説明するが、従来技術と重複する部分については説明を省略する。

【0049】いま、CPU1がOS10の管理のもとでAP32の1200番地を実行中であり、PC1aが次の1201番地を指している状態で外部装置3-1より割込要求があったものとする。PIC2がこれをCPU1に伝えると、CPU1は従来技術におけると同様、SP1bが指しているスタック領域33にPC1aの指している番地情報"1201"を中断するAP32の再開アドレスとして退避させる。これとともにSP1bはスタック領域33内の次の番地に進み、多重割込に備える。

【0050】次いで、CPU1は従来技術と同様、割込ベクタ領域21のタイプ0の番地にアクセスし、記憶されているアドレス情報"12340"を読み出す。これによってPC1aは12340番地の次の12342番地に移るが、前記の12340番地は従来技術と異なり、割込番号に対応する割込ハンドラ23の先頭番地ではなく、二重割込指定テーブル22の割込番号1に対応する番地である。

【0051】二重割込指定テーブル22は必要な割込番号の数だけ用意されるが、各割込番号に対応するメモリには同一内容、即ち、ソフト割込の命令が記憶されている。このソフト割込は外部装置よりの割り込みよりも優先度が高いものであるが、この割り込みも外部割込と同様、割込ベクタ領域21にアクセスして処理ルーチンが記憶されているメモリ領域の先頭番地を得るようになっている。ここではソフト割込の割込ベクタ領域21へのアクセスにタイプ255（物理アドレスは003FC）を使用することが機械語によって設定されている。

【0052】上記ソフト割込により二重割込が行われたことになるため、最初の割込でPC1aが移っていたアドレス値"12342"をSP1bが指しているスタック領域33に記憶し、SP1bは次の番地に移る。

【0053】ソフト割込が割込ベクタ領域21より得る情報、例えば"20000"は図示のようにOS10K一部を構成する割込ハンドラ起動処理部11の先頭番地となっている。割込ハンドラ起動処理部11は大別して、割込要因解析部11aとレジスタ保存処理部11bにより構成されるが、先のアドレス"20000"は割込要因解析部11aの先頭番地となっており、ここで割込要因の解析が行われる。

【0054】割込要因解析部11aには割込番号ごとの割込ハンドラの先頭番地が2バイト間隔のアドレスに記憶されているので、割込番号を算出すれば「20000+2×

(割込番号) - 2」の式により割込番号に対応する割込ハンドラのアドレスが判るようになっている。

【0055】前記割込番号は、スタック領域33に記憶したアドレス値"12342"から割込ベクタ領域21の先頭に記憶されているアドレス値の"12340"を引き、2で割ることにより得られる。即ち、割込要因の解析は簡単な演算を実行するのみで行われ、P I C 2の内部レジスタ(図示省略)にアクセスするが必要がないため、OSに大きなオーバーヘッドを生ずることがない。

【0056】この例では上記の演算により"1"が得られ、この割り込みが割込番号1であることが識別され、前記の式より割込ハンドラのアドレスが記憶されている番地が20000番地であることが求まる。この20000番地に記憶されているアドレス値を読むことにより割込ハンドラ31の処理に移ることができるが、それに先立ち、レジスタ保存処理部11bがCPU1の内部レジスタの保存を行う。

【0057】内部レジスタ類の保存処理は従来技術において割込ハンドラよりのシステムコールによって行う内容と同じであるが、本発明では割込ハンドラ31が関与せずにすべてOS内部のみで行うことができるため、ユーザがレジスタ類の退避に配慮して割込ハンドラを作成する必要がない。なお、このとき、データ・セグメントの付け替えなども行われる(詳細説明は省略)。

【0058】レジスタ保存処理部11bはレジスタの保存などが終わると割込ハンドラ31を呼び出すが、これは割込要因解析部11aが得た前記20000番地に記憶されているアドレス値を読むことにより行うが、アドレス値が例えば"3000"であれば、処理を3000番地に移すことにより割込ハンドラ31による処理が開始される。

【0059】処理が割込ハンドラ31に移されたのちにもし高位の割込要求があったときにはその割込処理に移る。そのため、OS10は割込ハンドラ31による割込処理中も割込状態即ち、現在の割込処理がどの外部装置よりの要求のものであるかを管理する必要があるが、これは先に割込要因解析部11aにより識別された割込番号をOS内部に保存することにより可能である。

【0060】以上のように、割込ハンドラ31はOSによって行われる処理がすべて終了したのちに起動されるため、CPU1の内部レジスタを保存したり、セグメントの付け替えのためにアセンブリ言語を使用したり、特別な手順を記述する必要がなく、割込処理内容のみを高級言語だけを用いて記述することができる。その結果、この割込ハンドラ31は通常のサブルーチンと同様に取り扱うことができる。

【0061】割込処理が終了する場合も割込ハンドラ31は内部レジスタやセグメントに関する特別な記述を行う必要がなく、サブルーチンからメインルーチンに戻す場合に通常使用される命令、例えばRETURN命令でOSに処理を戻すことができ、これによりOSは割込処理

が終了したことを知ることができる。

【0062】割込ハンドラ31よりの割込終了通知によりOSの割込ハンドラ完了処理部12はレジスタの復帰などの処理を行い、中断していたAP32の処理を再開させるが、その処理内容は従来技術とほぼ同一であるので説明を省略する。

【0063】図4は以上における処理の移行状態を図示したものであるが、図示のように、割込ハンドラは割込処理の際にCPUがレジスタの保存などの処理をすべて終了したのちに呼び出されるため、割込ハンドラ23の処理内容は非常に簡単になる。

【0064】図5は以上における割込ハンドラ起動時のCPUとOSの処理をフロー図に示したものであり、S1～S5はフローの各ステップを示す。割込が発生するとCPUのハード処理によって二重割込指定テーブルに無条件で分岐(S1)させたのち、二重割込指定テーブルにおいてタイプ255のソフト割込を発生(S2)させ、割込ハンドラ起動処理部において割込要因の解析(S3)とCPU内部レジスタの保存(S4)を行う。これらの処理が終わった時点で割込ハンドラを起動する(S5)。

【0065】割込ハンドラの処理終了時の細部の処理は前述のように従来技術と若干異なるが、フロー図上では差がないので説明を省略する。以上、図2乃至図5により本発明の実施例を説明したが、図2乃至図5はあくまで本発明の一実施例を示したものに過ぎず、本発明が図示したものに限定されるものでないことは勿論である。

【0066】例えば、上記の説明は8086シリーズのマイクロプロセッサを例として行ったが、本発明がマイクロプロセッサの名称如何に関わらず同様な構成をもつマイクロプロセッサを使用するマイクロプロセッサ・システムに適用可能であることは明らかである。

【0067】また、図3に示したメモリ構成におけるRAMの記憶内容の一部をROMに移しても本発明の効果が変わらないことは明らかであり、また、アドレスや記憶内容が図示のものに限定されないことは当然である。更に、OSが保存または付け替え処理を行う対象はレジスタやデータ・セグメントのみに限定されるものではなく、他の名称を有する同種の構成要素を含むことは勿論であり、本発明はマイクロプロセッサ・システムの構成の変形を排除するものではない。

【0068】

【発明の効果】以上説明したように、本発明においては、OSが内部レジスタ類の保存などの処理をすべて終了したのちに割込ハンドラを起動するため、割込ハンドラにレジスタの退避やデータ・セグメント付け替えのための手順を記述する必要がない。このため、ユーザは外部割込の処理内容を規定する割込ハンドラをすべて高級言語で記述することができ、かつ、記述の簡明化にともない誤処理が発生する可能性が減少する。また、割込要

因の解析が割込コントローラ（P I C）にアクセスすることなく簡単な演算によって行われるため、OSのオーバーヘッドが減少する。

【0069】即ち、本発明は、割込管理を行うマイクロプロセッサ・システムにおける割込ハンドラの作成効率と割込処理の効率の向上に大きく貢献する。

【図面の簡単な説明】

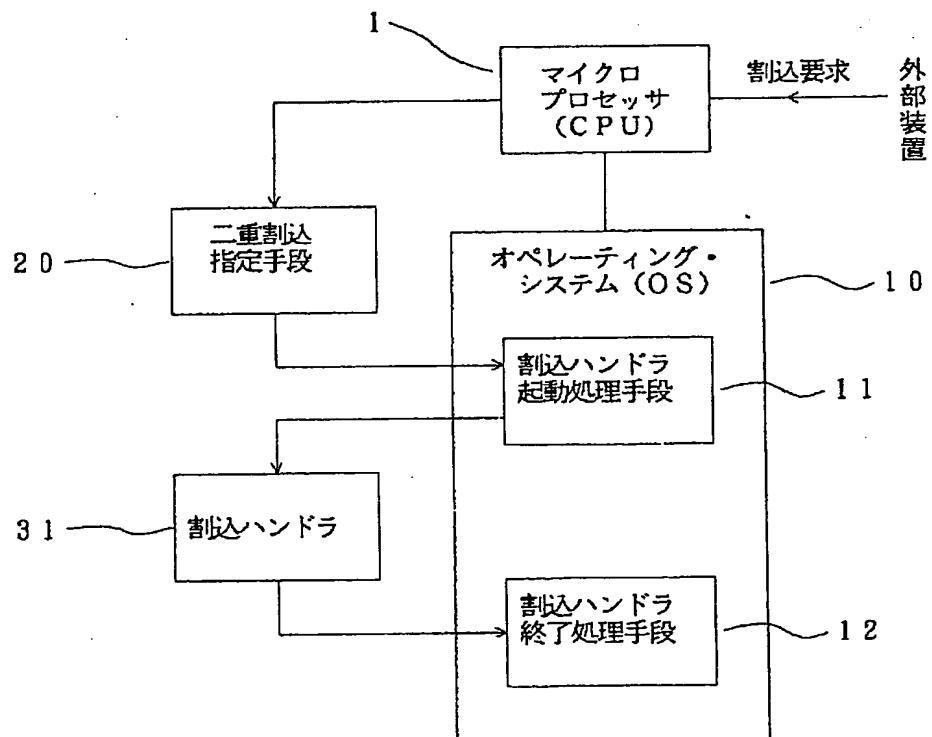
- 【図1】 本発明の基本構成図
- 【図2】 本発明の実施例システム構成図
- 【図3】 本発明の実施例メモリ構成図
- 【図4】 本発明の実施例処理移行状態説明図
- 【図5】 本発明の実施例フロー図
- 【図6】 マイクロプロセッサ・システム構成図
- 【図7】 従来技術のシステム構成図
- 【図8】 従来技術のメモリ構成図
- 【図9】 従来技術の処理移行状態説明図
- 【図10】 従来技術のフロー図

【符号の説明】

- 1 マイクロプロセッサ（C P U）
- 1a プログラムカウンタ（P C）
- 1b スタックポインタ（S P）
- 2 割込コントローラ（P I C）
- 3-1～3-8 外部装置（I/O）
- 10 オペレーティング・システム（O S）
- 11 割込ハンドラ起動処理手段
- 12 割込ハンドラ終了処理手段
- 13 実行管理部
- 20 二重割込指定手段
- 21 割込ベクタ領域
- 22 二重割込指定テーブル
- 31 割込ハンドラ
- 32 アプリケーション・プログラム（A P）
- 33 スタック領域

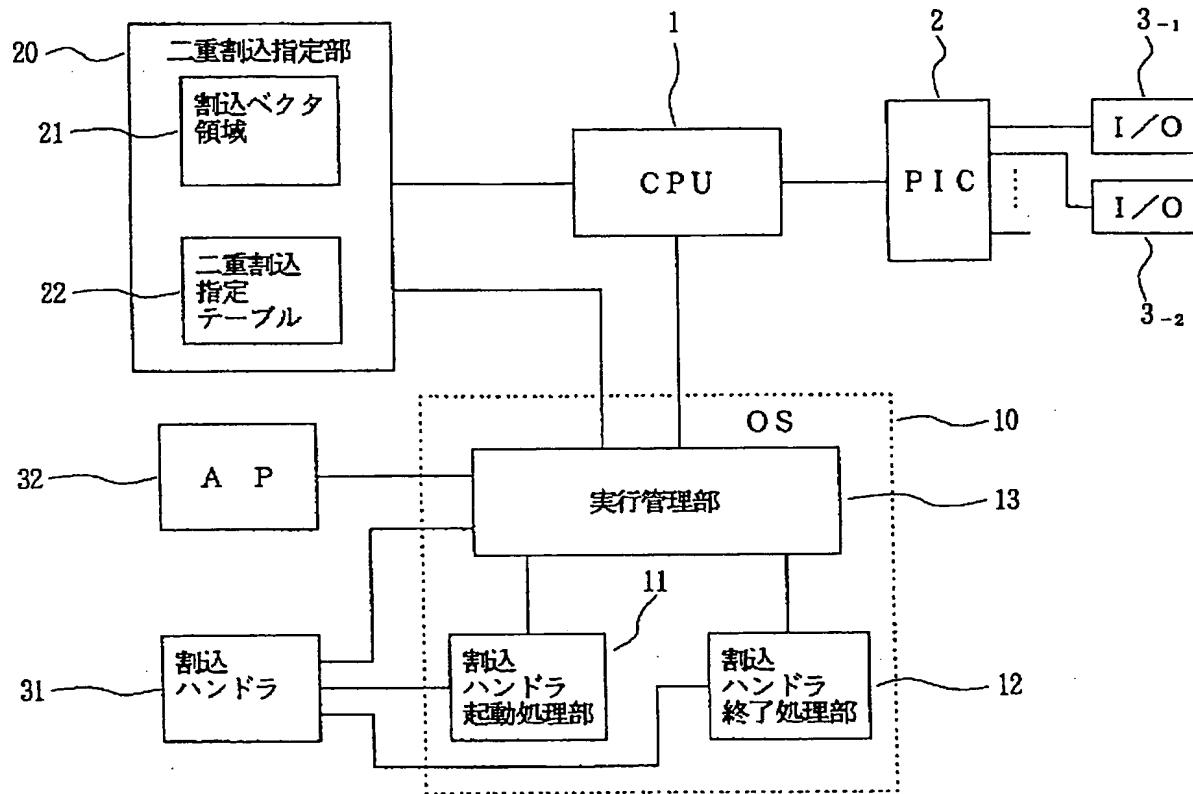
【図1】

本発明の基本構成図



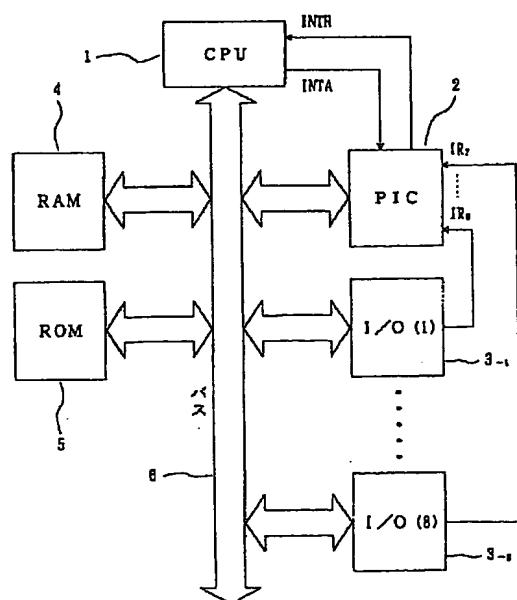
【図2】

本発明の実施例システム構成図



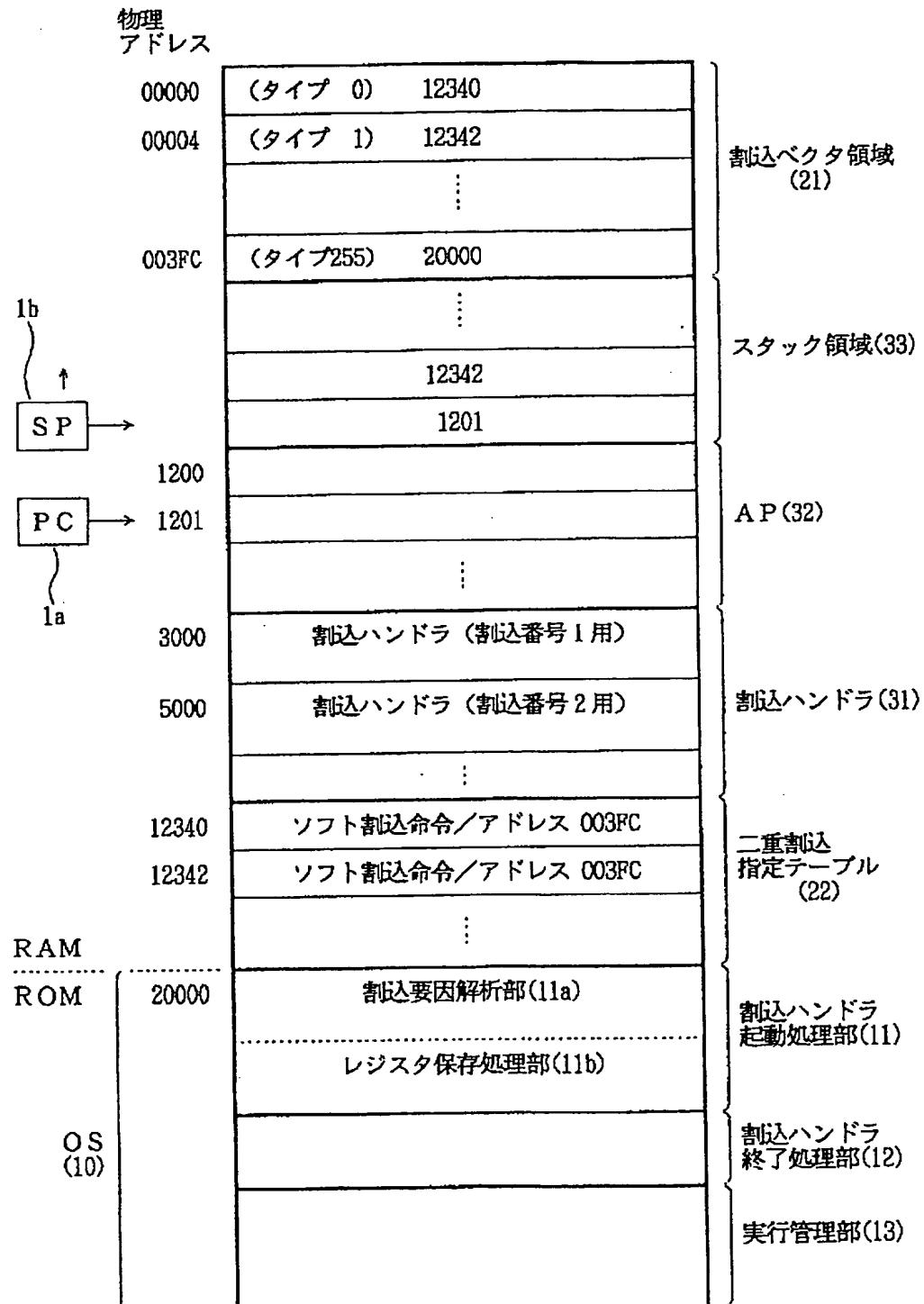
【図6】

マイクロプロセッサ・システム構成図



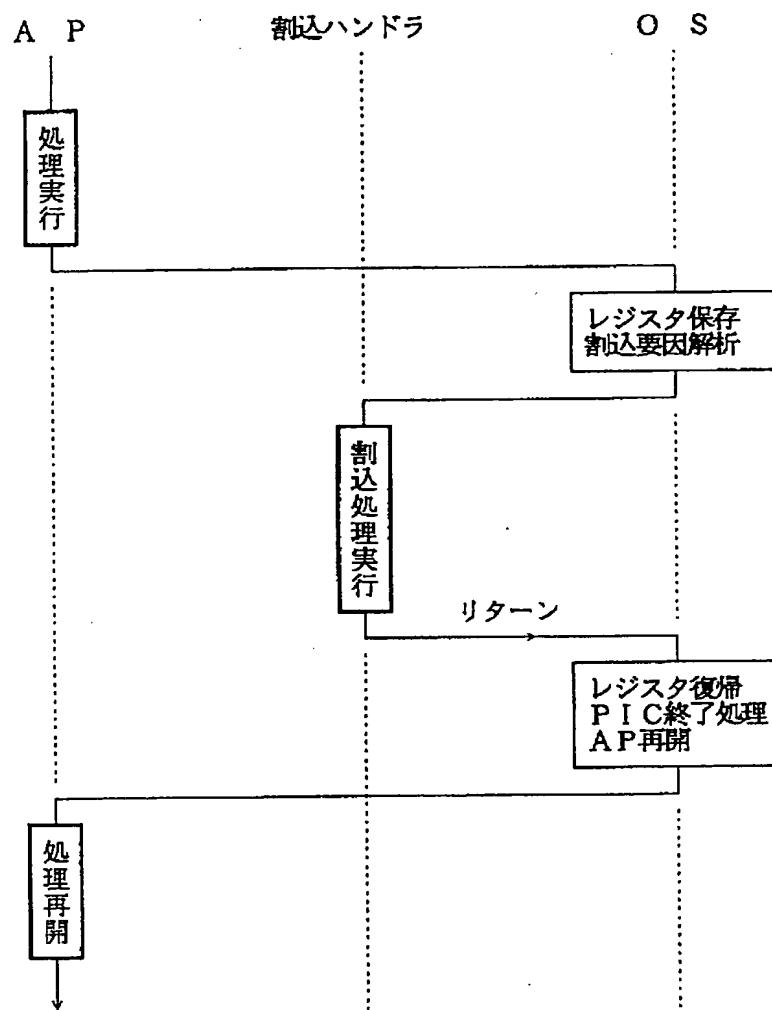
【図3】

本発明の実施例メモリ構成図



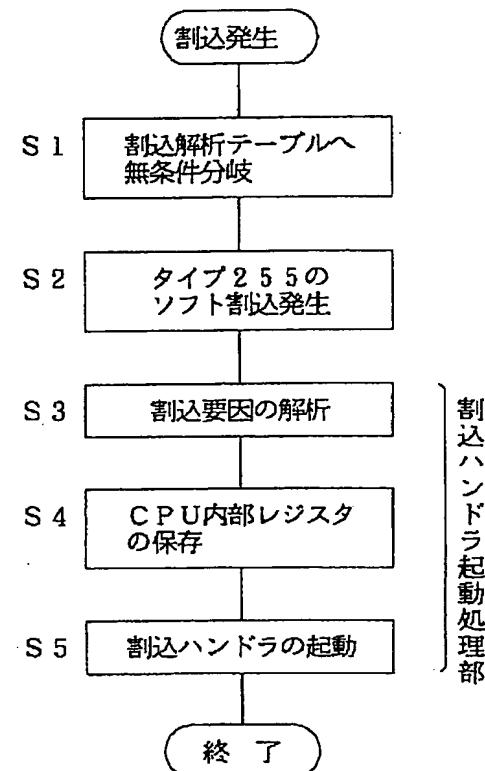
【図4】

本発明の実施例処理移行状態説明図



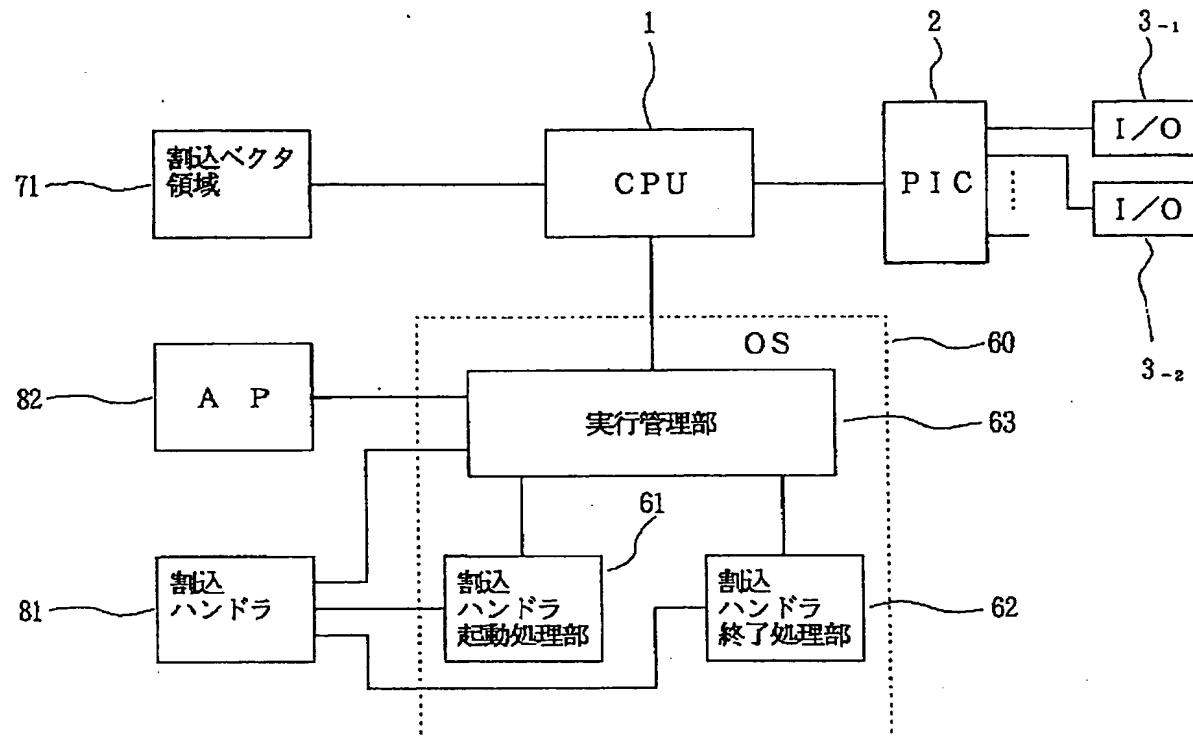
【図5】

本発明の実施例フロー図



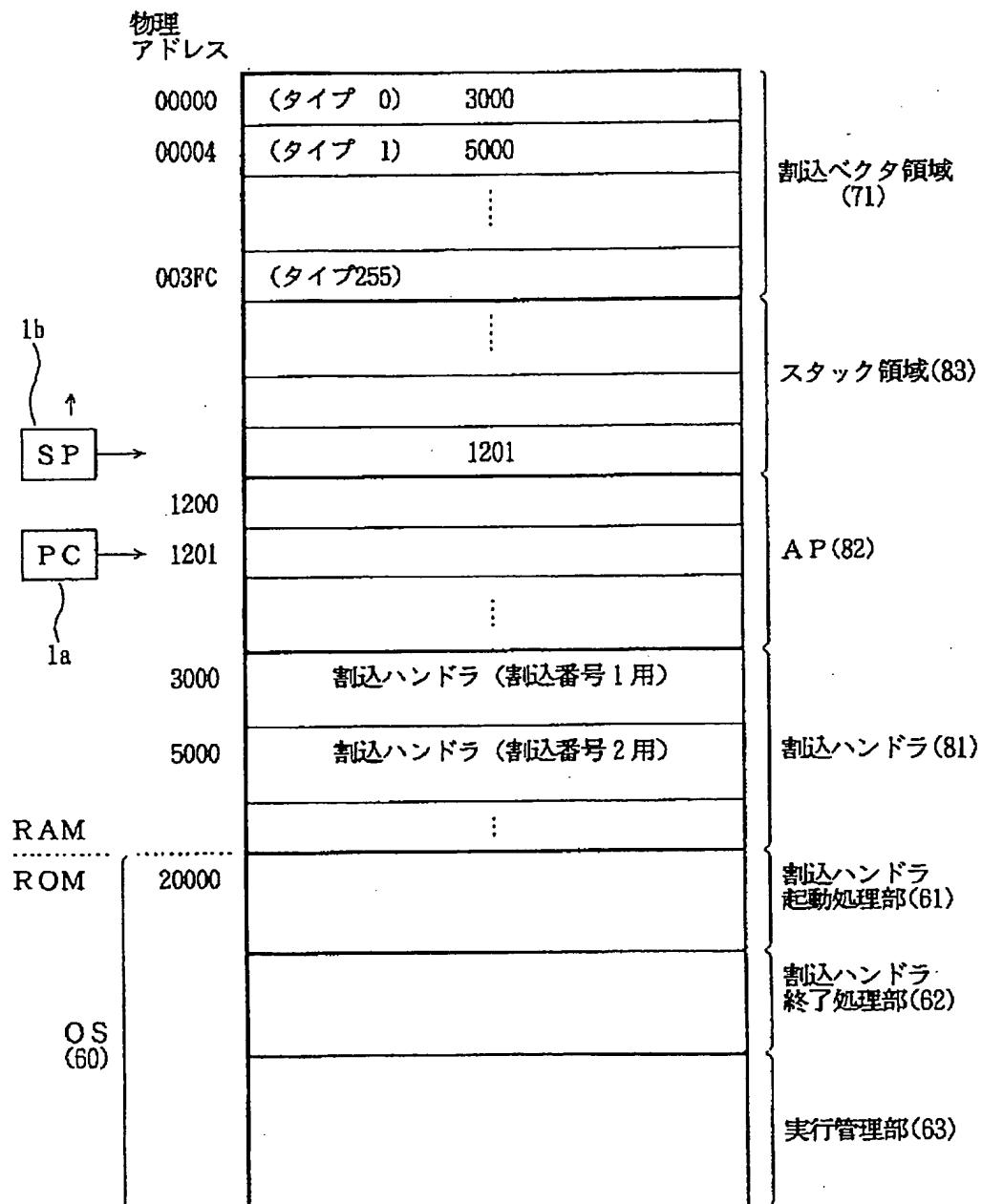
【図7】

従来技術のシステム構成図



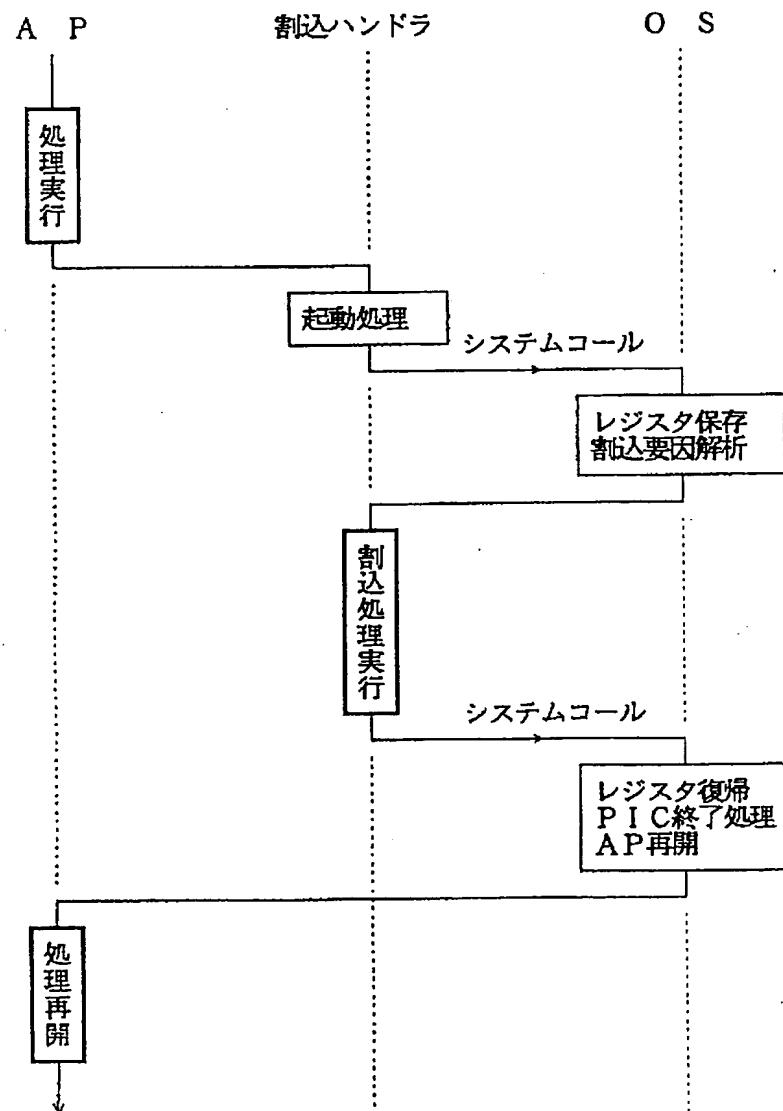
【図8】

従来技術のメモリ構成図



【図9】

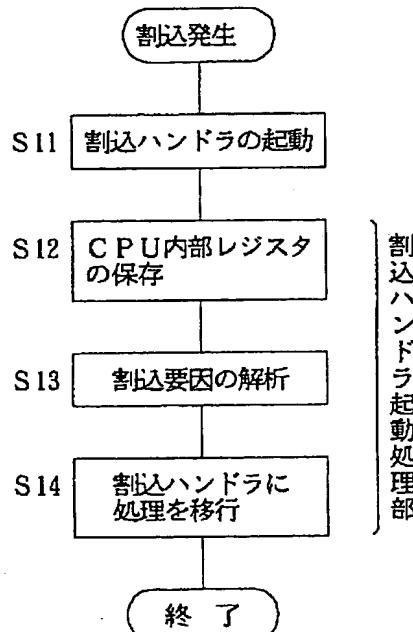
従来技術の処理移行状態説明図



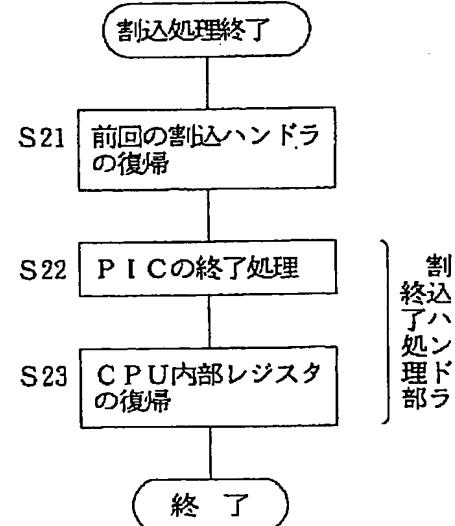
【図10】

従来技術のフロー図

(1) 割込ハンドラ起動時の処理



(2) 割込ハンドラ終了時の処理



フロントページの続き

(72)発明者 山脇 智洋

神奈川県横浜市港北区新横浜三丁目9番18
号 富士通コミュニケーション・システム
ズ株式会社内

(72)発明者 熱海 信夫

神奈川県横浜市港北区新横浜三丁目9番18
号 富士通コミュニケーション・システム
ズ株式会社内

(54) LANGUAGE PROCESSING PROGRAM EXECUTING DEVICE

(11) 5-224947 (A) (43) 3.9.1993 (19) JP

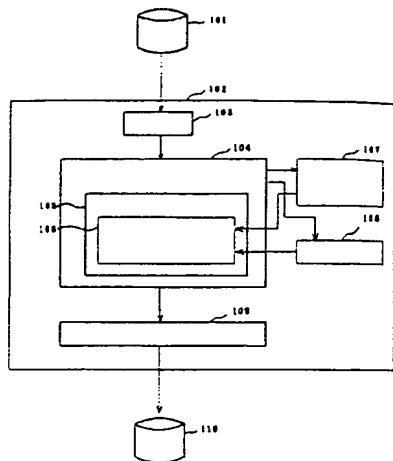
(21) Appl. No. 4-22886 (22) 7.2.1992

(71) NEC CORP (72) HIROKO ISOZAKI

(51) Int. Cl^s. G06F9/45, G06F9/46

PURPOSE: To curtail or delete the size for saving and reset of a internal variable area in pre-and post-processings of an interruption subroutine.

CONSTITUTION: The device is provided with a syntax analyzing part 104 for obtaining interruption subroutine information obtained by analyzing a source program information file containing the interruption subroutine information, and correlation information of a call of each subroutine, and an internal variable arrangement processing part 106 which is contained in the syntax analyzing part 104, and allocates an internal variable of each interruption subroutine and the subroutine called from the interruption subroutine into an exclusive high speed access memory of each interruption subroutine group. Accordingly, there is an effect for improving an execution time of a generated object and shortening the size.



101: source program information file, 102: language processing program executing means, 103: input part, 105: variable arrangement processing part, 107: interruption subroutine information, 108: correlation information, 109: object output part, 110: object file

(54) INTERRUPTION GENERATING CIRCUIT FOR DEBUGGING

(11) 5-224950 (A) (43) 3.9.1993 (19) JP

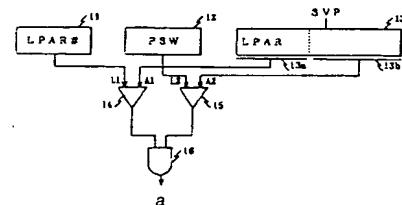
(21) Appl. No. 4-29201 (22) 17.2.1992

(71) HITACHI LTD(1) (72) KENICHI YAMAMURO(1)

(51) Int. Cl^s. G06F9/46, G06F11/28

PURPOSE: To efficiently execute a debug by providing a register (LPAR#) for holding a virtual computer number (LPAR number) in an instruction processor, and generating an interruption at the time of coincidence of the LPAR number being in the course of execution and an instruction address.

CONSTITUTION: A comparing circuit 14 compares the contents L1 of an LPAR# 11 and the contents A1 of a compare LPAR number 13a, and outputs "1" or "0" in accordance with a result of this comparison. On the other hand, a comparing circuit 15 compares the contents L2 of an instruction address in a program status word (PSW) 12 and the contents A2 of a compare instruction address 13b, and outputs "1" or "0" in accordance with a result of this comparison. Subsequently, an AND circuit 16 generates an interruption generating factor signal by AND of the output of the comparing circuit 14 and the output of the comparing circuit 15. This signal is supplied to an SVP 13 an interruption is generated. By storing a value required for the LPAR number 13a and the instruction address 13b, an interruption can be applied by limiting it to only one virtual computer.



a: interrupting signal

(54) INTERRUPTION MANAGING METHOD FOR MICROPROCESSOR SYSTEM

(11) 5-224951 (A) (43) 3.9.1993 (19) JP

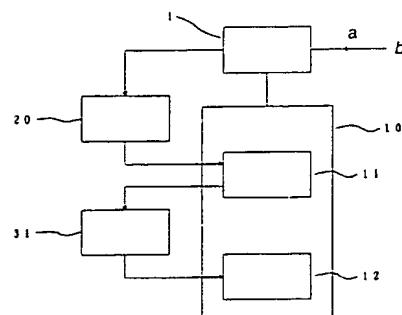
(21) Appl. No. 4-24226 (22) 12.2.1992

(71) FUJITSU COMMUN SYST LTD (72) YASUO NAKAJIMA(2)

(51) Int. Cl^s. G06F9/46, G06F13/24

PURPOSE: To provide the interruption managing method for generating easily an interruption handler, and decreasing an overhead of an OS.

CONSTITUTION: In the microprocessor system for suspending a processing at the time when a CPU 1 which is executing a processing through an OS 10, receives an interruption request from an external device, and allowing an interruption handler 31 to execute an interruption processing, this system is constituted by providing a double interruption designating means 20 for designating a soft interruption at the time when it is actuated from the CPU which receives the interruption request and migrating the processing to the OS before migrating the processing to the interruption handler, an interruption handler actuation processing means 11 for analyzing an interruption factor and retaining a register, etc., in the OS and actuating the interruption handler, and an interruption handler finish-processing means 12 for executing reset of the register, etc., and an interruption finish processing to the external device in the OS at the time when an end of the interruption processing is informed from the interruption handler, and thereafter, restarting the suspended processing.



a: interruption request, b: external device